IPST-CNAM
Systèmes répartis
**NFP 214**
Jeudi 12 Février 2009

Sans document
Durée : **2 heures**
Enseignants : LAFORGUE Jacques

1ère Session NFP 214

(COURS)

# 1. QCM (60 points)

Mode d'emploi :

Ce sujet est un QCM dont les questions sont de 3 natures :

- **les questions à 2 propositions**: dans ce cas une seule des 2 propositions est bonne.
  - +1 pour la réponse bonne
  - –1 pour la réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est bonne
  - + 1 pour la réponse bonne
  - –½ pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
  - + ½ pour chaque réponse bonne
  - –1 pour la réponse fausse

Il s'agit de faire une croix dans les cases de droite en face des propositions.

On peut remarquer que cocher toutes les propositions d'une question revient à ne rien cocher du tout (égal à 0).

Si vous devez raturer une croix, faites-le correctement afin qu'il n'y ait aucune ambiguïté.

N'oubliez pas d'inscrire en en-tête du QCM, votre nom et prénom.

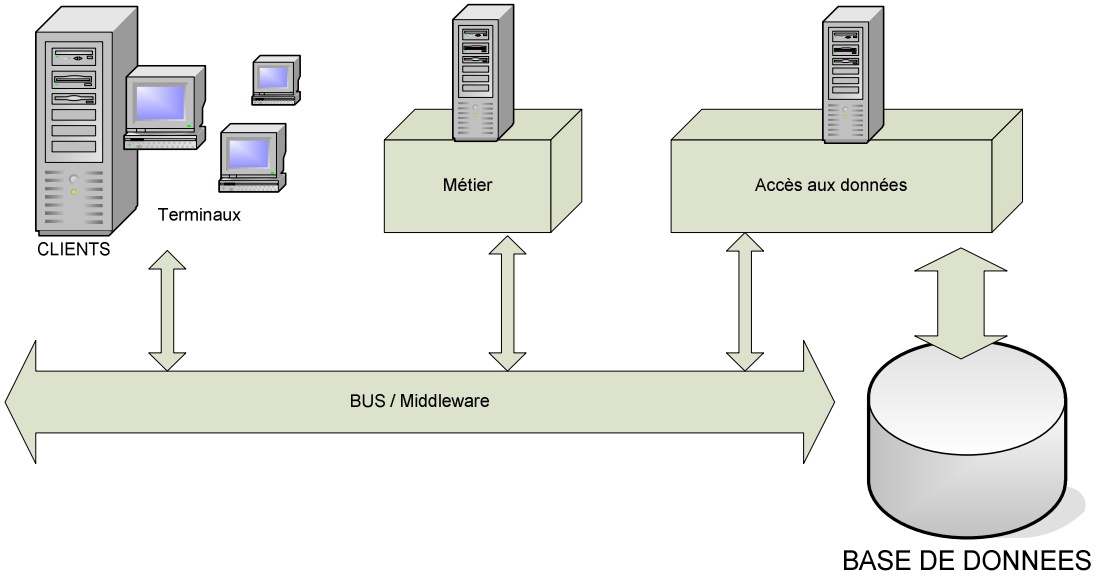
Vous avez droit à 10% de points négatifs sans pénalité.

| NOM: | PRENOM: |
|---|---|

| Un Middleware est : | | Q 1 |
|---|---|---|
| 1 | un framework de développement de solution client/serveur | |
| 2 | un intermédiaire de communication entre un client et un serveur | |
| 3 | un ensemble de composants logiciels assurant les interfaces de communication des données et l'appel aux traitements | |

| Dans une architecture serveur/client non répartie, le rôle d'un serveur est de centraliser l'information | | Q 2 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| La communication logique entre les clients et le serveur | | Q 3 |
|---|---|---|
| 1 | permet l'échange d'informations logiques (au sens de la compréhension mutuelle des échanges) | |
| 2 | d'abstraire au maximum les mécanismes d'échange des informations | |

| Le schéma ci-dessous est une représentation d'une architecture 3-Tiers: | | Q 4 |
|---|---|---|
|  | | |
| 1 | OUI | |
| 2 | NON | |

| Une architecture distribuée est : | | Q 5 |
|---|---|---|
| 1 | une architecture dans laquelle toutes les ressources sont centralisées dans un "chef d'orchestre" (élément central indispensable) | |
| 2 | une architecture dans laquelle les ressources sont réparties sur le réseau | |
| 3 | une architecture dans laquelle certaines ressources sont réparties dans des factorys | |

| On oppose habituellement le système réparti au système centralisé.<br>Y a t-il un avantage qu'un système réparti se comporte comme un système centralisé ? | | Q 6 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un objet distribué est une donnée informatique qui est échangée entre un client et son serveur | | Q 7 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| RMI (Remote Méthod Invocation) est un standard de communication entre des composants Java | | Q 8 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| CORBA (Common Object Request Broker Architecture) est | | Q 9 |
|---|---|---|
| 1 | une API exclusivement Java permettant la communication de données à travers un réseau | |
| 2 | une norme de Middleware dédié à la communication de données à travers un réseau | |
| 3 | un Framework Java et C++ de développement de solution informatique orientée serveur/client | |

| Un Factory est | | Q 10 |
|---|---|---|
| 1 | une usine de fabrication d'objets distribués | |
| 2 | une usine de fabrication de liens de persistance entre les objets et leurs représentations en base de données | |
| 3 | une usine de fabrication de servant (CORBA) | |

| Un objet distribué est toujours nommé et enregistré dans un annuaire (ou adaptateur). | | Q 11 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

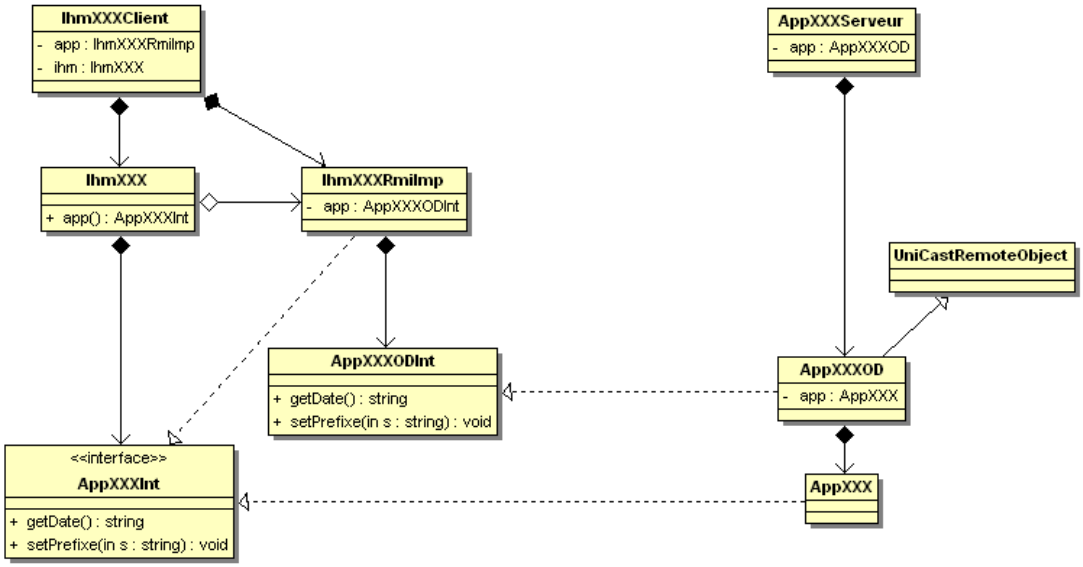
| Soit, un IDL qui contient une interface de nom *InterfaceDeviseOD*. Cette interface génère, en Java, une classe utilisée par héritage pour créer les servants. Le nom de cette classe est : | | Q 12 |
|---|---|---|
| 1 | InterfaceDeviseODOperations | |
| 2 | InterfaceDeviseODPOA | |
| 3 | InterfaceDeviseODHolder | |

| Soit, un IDL qui contient une interface de nom *InterfaceDeviseOD*. Cet IDL génère, en Java, une interface utilisée pour implémenter les méthodes distantes. Le nom de cette interface est : | | Q 13 |
|---|---|---|
| 1 | InterfaceDeviseODOperations | |
| 2 | InterfaceDeviseODPOA | |
| 3 | InterfaceDeviseODHolder | |

| Les systèmes répartis sont tout naturellement des systèmes hautement sécurisés aux intrusions extérieures. | | Q 14 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| L'indépendance de la situation géographique d'un objet distribué passe par la mise en place d'un "chef d'orchestre" qui mémorise les paramètres de connexion de cet objet distribué. | | Q 15 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| La tolérance aux pannes d'un servant dans une architecture répartie passe, notamment, par | | Q 16 |
|---|---|---|
| 1 | la mise en place du doublement du réseau | |
| 2 | la mise en place de l'indépendance géographique du servant pour les utilisateurs | |
| 3 | la mise en place de liens de réplication des ressources critiques du servant sur un autre servant | |




| | | Q 17 |
|---|---|---|
| Cette représentation UML est celle de l'atelier 16 dans le cas d'une solution RMI. Elle décrit la dépendance des classes et des interfaces entre un client IHM et un serveur RMI.<br>Le rôle de la classe IhmXXXRmiImp est essentiel car : | | |
| 1 | Elle encapsule l'applicatif permettant ainsi à l'IHM d'utiliser les méthodes de l'applicatif | |
| 2 | Elle implémente l'interface AppXXXInt permettant ainsi à l'IHM d'utiliser les méthodes de l'applicatif comme si elles étaient distantes via une communication RMI | |
| 3 | Elle utilise l'interface AppXXXODInt qui communique avec le serveur via l'amorce du bus RMI. | |

| Les rôles d'un ORB sont notamment : | | Q 18 |
|---|---|---|
| 1 | de ne pas rendre transparente la localisation des objets distribués | |
| 2 | de faciliter le passage des paramètres dans les appels de méthodes distantes | |
| 3 | de permettre l'échange de référence d'objet distant (ou proxy) | |

| Pour utiliser les méthodes distantes d'un objet distribué : | | Q 19 |
|---|---|---|
| 1 | on demande à un adaptateur le proxy de connexion de l'objet distribué | |
| 2 | on demande à un factory le proxy de connexion de l'objet distribué | |
| 3 | on demande à un IDL le proxy de connexion de l'objet distribué | |

| Un proxy de connexion (ou référence d'un objet distribué) est un pointeur sur un objet distribué | | Q 20 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un proxy de connexion (ou référence d'un objet distribué) est un pointeur sur l'amorce (ou stub) | | Q 21 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Deux clients peuvent utiliser en même temps une méthode distante d'un objet distribué | | Q 22 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

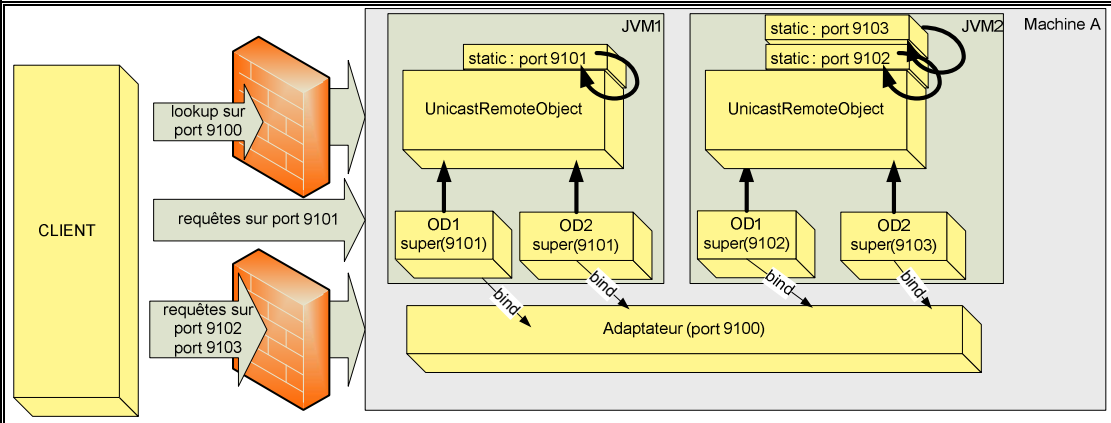
| On peut interdire que deux clients puisent utiliser en même temps une méthode distante d'un objet distribué | | Q 23 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En CORBA, l'IOR est créé par le client, la première fois, lors de sa connexion au serveur | | Q 24 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Il existe 4 niveaux hiérarchiques de couche de communication dans un système réparti | | Q 25 |
|---|---|---|
| 1 | Envoi de message, Appel de fonction, Appel de méthode, Appel d'un service | |
| 2 | POA, Servant, Thread, Socket | |
| 3 | Serveur, Agent, Thread, Socket | |

| En RMI, le client décide de la manière de communiquer avec le serveur (choix de la connexion socket) | | Q 26 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En RMI, le Socket Factory est géré par la classe UnicastRemoteObject | | Q 27 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

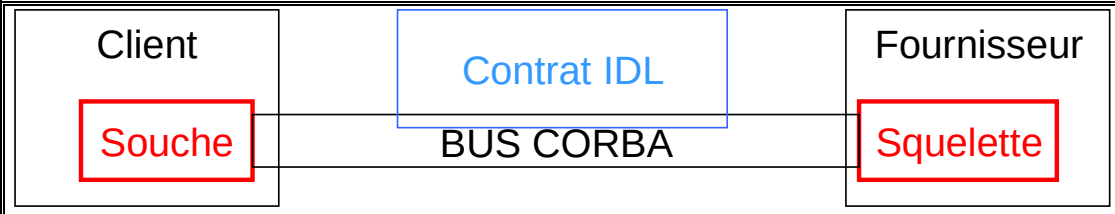
|  <br>Ce schéma représente le fonctionnement des "Sockets factorys" des classes UnicastRemoteObject utilisées dans des programmes Java RMI. | | Q 28 |
|---|---|---|
| 1 | Les ports des Sockets Factorys sont ici déterminés par le système | |
| 2 | Les ports des Sockets Factorys sont ici imposés par le développeur | |
| 3 | Dans la JVM2 du schéma, on pourrait créer un nouvel objet distribué RMI sur le port 9102 | |

| Une communication de l'appel d'une méthode distante, entre un client et un objet distribué RMI se fait : | | Q 29 |
|---|---|---|
| 1 | directement entre le client et l'objet distribué | |
| 2 | entre le client er l'objet distribué en utilisant un stub et un skeleton mais sans passer par l'adaptateur | |
| 3 | entre le client er l'objet distribué en passant par l'adaptateur | |

| En CORBA, le POA est le composant qui permet d'enregistrer un servant dans le service de nommage | | Q 30 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En CORBA, le service de nommage est créé sur chaque machine du réseau | | Q 31 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Cette instruction :<br>*POA poa = POAHelper.narrow( orb.resolve_initial_references( "RootPOA" ));*<br>consiste à récupérer une interface de connexion avec le POA par défaut d'un bus CORBA | | Q 32 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |




| En CORBA, le contrat IDL d'un bus CORBA : | | Q 33 |
|---|---|---|
| 1 | est un programme IDL contenant la description des interfaces qui précisent les méthodes distantes du Fournisseur, utilisées par le Client | |
| 2 | est l'ensemble de classes Java, C++, ... générées par les pré-compilateurs Idl vers Java, Idl vers C++, ....qui contiennent la souche et le squelette | |
| 3 | est un programme d'interface entre le Client et le Fournisseur, enregistré dans le service de nommage | |

| En Java RMI, l'instruction **lookup** : | | Q 34 |
|---|---|---|
| 1 | retourne un stub | |
| 2 | retourne un skeleton | |

| En Java RMI, l'instruction **bind** : | | Q 35 |
|---|---|---|
| 1 | permet de créer un objet distribué | |
| 2 | alloue un port de communication pour l'objet distribué | |
| 3 | permet d'enregistrer un objet distribué dans un adaptateur | |

| Toutes les méthodes distantes d'un objet distribué RMI | | Q 36 |
|---|---|---|
| 1 | doivent appartenir à la même interface | |
| 2 | peuvent appartenir à plusieurs interfaces | |
| 3 | doivent toutes être implémentées dans le même objet distribué | |

| On peut passer en paramètre d'une méthode distante, un objet distribué afin que le client puisse l'utiliser pour ensuite utiliser les méthodes distantes de cet objet distribué | | Q 37 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Les missions du Middleware est de : | | Q 38 |
|---|---|---|
| 1 | implémenter la couche de transport (couche 5, 6 et 7) | |
| 2 | gérer l'interface de communication entre le serveur et la couche de transport | |
| 3 | gérer les appels distants et la mise en forme des données | |

| Le Domain Name System (DNS) n'est pas un système réparti mais un système à plat de l'ensemble des adresses des domaines et centralisé sur un serveur unique appelée DNSS (Domain Name System Server) | | Q 39 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Une architecture 3-Tiers est une architecture 3 Couches (ou Niveaux) constitué des couches de Présentation, de Traitement et d'Accès aux données | | Q 40 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En CORBA, sur l'appel d'une méthode distante, il est possible de faire passer en paramètre un objet contenant du code. | | Q 41 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Au sujet de l'utilisation d'un objet distribué (OD) sur Internet : | | Q 42 |
|---|---|---|
| 1 | Rien n'interdit qu'un OD puisse être utilisé par des clients connectés sur Internet. | |
| 2 | Pour utiliser un OD sur Internet, il faut sécuriser les ports gérés par l'OD et l'adaptateur | |
| 3 | Il est impossible qu'un OD puisse être utilisé par des clients connectés sur Internet. | |

| Pour utiliser un OD sur Internet, il vat mieux les intégrer dans une architecture 4-Tiers (exemple : Internet Explorer / Tomcat / EJB RMI / BD) | | Q 43 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Une application répartie est une application asynchrone | | Q 44 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Pour mette en place une redondance chaude d'un nœud d'un système réparti RMI, il faut : | | Q 45 |
|---|---|---|
| 1 | Une fois que les clients ont récupéré (lookup) la connexion au nœud, enregistrer le nœud redondant sous le même nom.<br>En cas de perte de connexion, les clients doivent refaire le lookup sur le même nom.<br>Un lien de réplication doit être établi entre le nœud maître et le nœud esclave | |
| 2 | Dès la création du nœud maitre, il faut l'enregistrer dans l'adaptateur et le re-enregistrer afin que l'adaptateur en garde une copie et maintienne le lien de réplication entre le nœud maître et le nœud esclave | |

| Les modèles informatiques suivants sont des modèles d'architecture répartie : | | Q 46 |
|---|---|---|
| 1 | DNS (Domain Name System) | |
| 2 | Sockets | |
| 3 | Systèmes multicouches | |

| En RMI, le code suivant est un exemple de création d'un objet distribué :<br>*HelloOD od = new HelloOD("Pierre","DUPONT");*<br>*UnicastRemoteObject.exportObject(od);*<br>*Naming.rebind("rmi://localhost:9999/HELLO",od);* | | Q 47 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Une interface RMI hérite de l'interface RemoteException | | Q 48 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Dans le protocole RMI la communication entre deux objets distribués se fait en utilisant la technologie des sockets | | Q 49 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un objet passé en paramètre d'une méthode distante est reçu par l'appelant : | | Q 50 |
|---|---|---|
| 1 | sous la forme d'un nouvel objet qui est une copie du paramètre | |
| 2 | sous la forme d'une référence d'un objet distant | |

| Pour pouvoir faire communiquer un composant Java et un composant C++ sur un bus CORBA | | Q 51 |
|---|---|---|
| 1 | on doit exécuter les deux composants sur deux machines différentes | |
| 2 | on doit créer un socket de communication entre les deux composants | |
| 3 | on doit définir un IDL | |

| En JAVA, la persistance des objets en base de données se fait en utilisant la réflexivité du langage | | Q 52 |
|---|---|---|
| 1 | OUI | |

| | | |
|---|---|---|
| 2 | NON | |

| Les IOR sont des chaînes de caractère uniques sur un réseau donné permettant de créer une référence CORBA | | Q 53 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| En RMI, un client accède au contenu de certains attributs d'un objet distribués.<br>Pour y accéder : | | Q 54 |
|---|---|---|
| 1 | les attributs doivent être publics | |
| 2 | les attributs doivent être privés | |
| 3 | les attributs sont accessibles via des méthodes distantes | |

| Une méthode distante d'un objet distribué RMI peut être static | | Q 55 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| La sérialisation est le seul moyen pour passer des paramètres dans les appels des méthodes distantes d'un objet distribué | | Q 56 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| La sérialisation est utilisé, notamment, pour : | | Q 57 |
|---|---|---|
| 1 | transporter les données entre un client et un serveur | |
| 2 | faire la sauvegarde de données sur disque | |
| 3 | structurer les données de l'interface IDL qui définie les données échangées entre le client et le serveur | |

| L'IDL est un langage informatique permettant de décrire le contrat d'échange (données et traitements) entre les clients et leur serveur | | Q 58 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un client qui appelle une méthode distantes synchronisée, interdit à tout autre client d'utiliser l'objet distribué tant qu'il n'a pas fini d'exécuter la méthode | | Q 59 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

| Un adaptateur ou service de nommage utilise un port de la machine sur lequel il se met en attente de demande de connexion des clients de l'objet distribué qui s'y est enregistré. | | Q 60 |
|---|---|---|
| 1 | OUI | |
| 2 | NON | |

# 2. Questions libres (20 points)

Chaque question est notée sur 4 points.
*Vous répondez à ces questions sur une **copie vierge** en mettant bien le numéro de la question, sans oublier votre nom et prénom.*

**Q 1**

Les propriétés de transparence d'un système réparti sont:

- transparence d'accès et de localisation
- transparence du partage des ressources
- transparence de la réplication
- transparence des fautes
- transparence de la migration
- transparence de charge
- transparence d'échelle

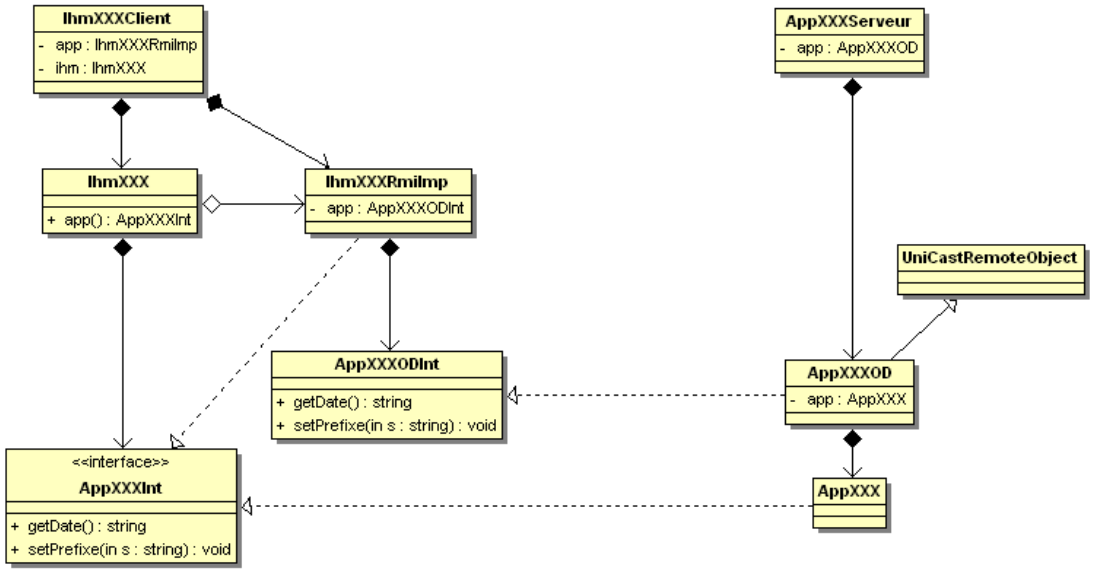
Donnez la signification de chacune de ces transparences.

**Q 2**

1/Expliquez pourquoi les ORB bâtis sur le modèle d'objet distribué tel que RMI ou CORBA, sont mal adaptés à Internet.
2/Décrivez en 2 ou 3 phrases et 1 shéma, le modèle d'architecture qui permet d'apporter des solutions aux problèmes que vous avez cités à la question juste précédente (Q2-1/).

**Q 3**

Ce schéma est la description statique UML des classes et interfaces mise en œuvre pour la conception d'un client avec son serveur RMI composé d'un objet distribué.




Indiquez le rôle de chacune de ces classes et/ou interfaces Java. (Précisez bien quand c'est une classe ou une Interface).

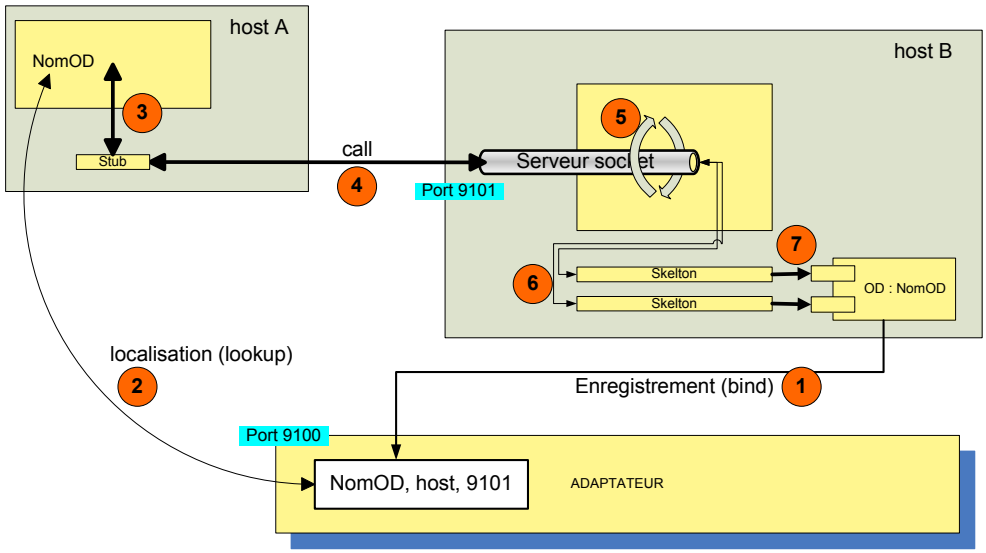
**(Tourner la page)**

**Q 4**

Citez des avantages de l'utilisation de la norme CORBA.

**Q 5**

Ce schéma est la description de l'architecture dynamique de la mise en oeuvre de la communication d'un client avec un Objet Distribué.




Commentez chacun des points numérotés.

***(Fin du sujet)***